Wire Detection Algorithms for Navigation

NASA Research Grant NAG2-1487
for the period July 1, 2001 to June 30, 2002
The Final Report Submitted To
NASA Ames Research Center
Grants Officer, N241-1
Moffett Field, California 94035-1000
NASA Technical Contact: Albert Ahumada

by

Rangachar Kasturi and Octavia I. Camps
Principal Investigators
Department of Computer Science and Engineering and
Electrical Engineering
The Pennsylvania State University
University Park, PA 16802
Tel: (814) 863-4254 and (814) 863-1267
E-mail: kasturi,camps@cse.psu.edu

Students:
Ying Huang
Anand Narasimhamurthy
Nitin Pande

Abstract

In this research we addressed the problem of obstacle detection for low altitude rotorcraft flight. In particular, the problem of detecting thin wires in the presence of image clutter and noise was studied. Wires present a serious hazard to rotorcrafts. Since they are very thin, their detection early enough so that the pilot has enough time to take evasive action is difficult, as their images can be less than one or two pixels wide.

Two approaches were explored for this purpose. The first approach involved a technique for sub-pixel edge detection and subsequent post processing, in order to reduce the false alarms. After reviewing the line detection literature, an algorithm for sub-pixel edge detection proposed by Steger was identified as having good potential to solve the considered task. The algorithm was tested using a set of images synthetically generated by combining real outdoor images with computer generated wire images. The performance of the algorithm was evaluated both, at the pixel and the wire levels. It was observed that the algorithm performs well, provided that the wires are not too thin (or distant) and that some post processing is performed to remove false alarms due to clutter.

The second approach involved the use of an example-based learning scheme namely, Support Vector Machines. The purpose of this approach was to explore the feasi-

bility of an example-based learning based approach for the task of detecting wires from their images. Support Vector Machines (SVMs) have emerged as a promising pattern classification tool and have been used in various applications. It was found that this approach is not suitable for very thin wires and of course, not suitable at all for sub-pixel thick wires. High dimensionality of the data as such does not present a major problem for SVMs. However it is desirable to have a large number of training examples especially for high dimensional data. The main difficulty in using SVMs (or any other example-based learning method) is the need for a very good set of positive and negative examples since the performance depends on the quality of the training set.

Contents

List of Figures

1 Introduction

Continued advances in the fields of image processing and computer vision have raised interest in their suitability in aiding pilots to early detection of possible obstacles in their flight trajectories. For example, as part of the High Speed Research (HSR) Vision program at the NASA Ames Research Center, we have recently studied, designed, and tested a computer vision system capable of real-time obstacle detection during mid-flight of an aircraft [6, 15]. Before that, obstacle detection on runways during take-offs and landings was also studied [14].

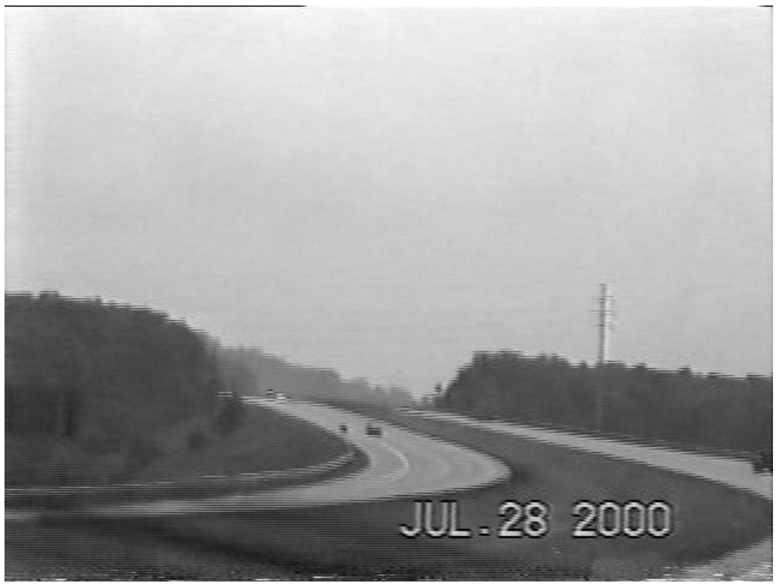


Figure 1: Thin wire with clouds in the background, and noise due to camera jitter.

Some of our previous results can be naturally extended to obstacle detection for low-altitude flight of helicopters. However, the fact that the rotorcraft is close to the ground for most of the time, places more severe requirements to the algorithms to be used. For example in the image shown in figure 1, the system must deal with multiple ground-based obstacles such as wires, trees, etc. in the presence of severe camera jitter and ever present cluttered background due to the ground. Of these obstacles, the most difficult to detect are wires since they are very thin and their image from the rotorcraft can be less than a pixel wide.

In this report we describe a preliminary study on the use of a line detection algorithm to detect wire obstacles in the path of rotorcrafts flying at low altitudes. Two approaches were explored. The first approach involved the use of an algorithm capable of detecting thin lines (sub-pixel) and subsequent post processing. An algorithm proposed by Steger [22] and a Hough transform to eliminate false alarms were identified as good candidates for this task since they are capable of line detection with sub-pixel accuracy. The algorithms are described in section 3. Due to the fact that real data was not available to test the algorithms, a set of testing data was generated by combining natural background images with computer generated wires, corrupted with synthetic noise. The procedure used to generate the data is described in section 4. In order to evaluate the performance of the algorithm a set of experiments were conducted for different sizes of wires at various distances. The experimental protocol used in these experiments is described in section 5 and the obtained results are summarized in section 6.

The second approach explored the feasibility of an example based learning method for the particular task. Support Vector Machines (SVMs) were used in this approach. SVMs have emerged as an important pattern classification tool in the recent past and have been used in various applications. They have a number of advantages as compared to other example based learning techniques such as neural networks. SVMs are briefly described in section 7 and an overview of SVM theory is presented in Appendix A. The mathematical details are presented in Appendix B. The interested reader is referred to [26] for a comprehensive treatment of learning theory. [1] is a good tutorial on SVMs. [19] contains a lucid and a somewhat detailed treatment of the theory of SVMs. The implementation details are discussed in section 8 and the training of the SVM is described in section 8.1. Preliminary results are presented in section 9.

Finally, the conclusions and directions for future research are discussed in section 10.

2 Needs and Requirements

NASA's need for enhanced capabilities in obstacle detection using image processing requires robust, reliable and fast techniques. Low-altitude rotorcraft navigation must often avoid ground-based obstacles such as electric wires, antennas, poles, trees and buildings. Electric wires are very thin objects and hence their images can have sub-pixel thickness. On the other hand, trees and buildings typically occupy several pixels. Furthermore, low-altitude flight implies, in general, severe background clutter due to the ground. Thus, the obstacle detection techniques should provide a high probability of *timely* detection while maintaining a low probability of false alarm in noisy, cluttered images of obstacles exhibiting a wide range of sizes and complexities. Moreover, these techniques should work well under the controlled conditions found in a laboratory and with data closely matching the hypothesis used in the design process, but it must be insensitive – i.e. must be *robust* – to data uncertainty due to various sources, including sensor noise, camera jitter, weather conditions, and cluttered backgrounds.

3 Wire Detection using Computer Vision

Electric wires between poles hang forming catenary curves and wires holding hanging bridges hang forming parabola curves. However, for detection purposes these curves, can be approximated as piece-wise linear. Thus, for this study we have confined ourselves to the problem of line detection in cluttered images. Furthermore, since wires are thin and their images from a far enough distance are typically less than a pixel wide, we paid special attention to algorithms that could provide sub-pixel accuracy.

3.1 Line Detection Algorithms

Detection of curvilinear and piece-wise linear structures in gray scale images has a wide range of applications including medical imaging, remote sensing, photogrammetry and line drawing understanding and has been the focus of much attention in computer vision research. Next, we present a brief overview of line detection techniques. For more details see for example [11].

- **Edge detection based approaches**

  Lines can be detected by locating "edges" – i.e. pixels where the image gray levels undergo large variations. Thus, most edge-based line detection techniques rely on operators approximating the image gradient. Examples of this approach are the Roberts, Prewitt, and Sobel operators where the image gradient components are approximated as weighted averages of gray level differences in the pixel neighborhood, computed using a pair of small masks. Edges are then found by thresholding the magnitude of the image gradient. This procedure typically results in "thick edges" that must be thinned and gaps that must be closed using a cleaning procedure.

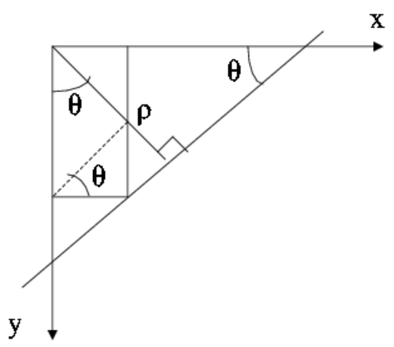


Figure 2: Normal representation of the line

Alternatively, edges can be located by finding the zero-crossings of the image Laplacian. Since the second derivative operator is very sensitive to noise, the image Laplacian is usually applied in conjunction with a noise filtering such as a Gaussian filter.

Probably, the edge detector most commonly used today is the Canny edge operator which uses first derivative of Gaussian filters to closely approximate the operator with optimal signal-to-noise ratio and edge localization.

- **Hough transform based approaches**

  Lines and curves can be found by linking adjacent edges into contours. The Hough transform was introduced to detect complex patterns and quickly adapted to detect lines and curves. The main idea of the Hough transform is to map the pattern detection problem into the easier problem of detecting a peak in the space defined by a set of parameters describing the pattern being sought. Consider for example, a line expressed using its normal representation (see figure 2):

$$\rho = x \cos\theta + y \sin\theta$$

where ρ represents the distance between the image origin and the line and θ is the line orientation. Each edge pixel (x_p, y_p) constrains the set of possible pairs of parameters (ρ, θ) of lines containing the edge by the sinusoidal expression:

$$\rho = x_p \cos\theta + y_p \sin\theta$$

Collinear edges share an unique pair (ρ, θ) which must satisfy all the constraints and thus corresponds to the point in parameter space where all the associated constraints intersect. The longer the line, the more edges sharing the same pair of parameters and the larger the number of constraints intersecting at one point in parameter space. Thus, lines can be found by discretizing the parameter space, associating to each cell a counter of the number of constraints passing through the cell, and finding peaks among the counter values.

- **Curve fitting based approaches**

  Splines are widely used to represent curves. Although splines can be made by joining any kind of function end to end, the most commonly used splines use piecewise cubic polynomials. Cubic polynomials provide enough degrees of freedom to determine edge location and orientation. Algorithms for edge detection using B-splines are described in [3] and [8].

- **Detection of thin lines**

  Thin lines can be detected by modeling them as objects with parallel edges [5, 7] and using a pair of edge detector filters to find the left and right edges of the line or by using differential geometry properties to find ridges and ravines on the image surface $z(x, y)$ [17, 16, 4, 2, 9]. Recently, Steger [22] proposed a detection algorithm based on differential geometry capable of detecting lines with sub pixel accuracy. He applied the algorithm to detect of roads from satellite images and to detect very thin lines in MR and angiogram medical images. Steger's algorithm was capable of detecting lines at different scales, even in the presence of severe clutter. Furthermore, the algorithm retrieved the precise line locations (defined as their median axis) and the line widths with sub pixel accuracy. Due to the quality of these results, and the similarity between the complexity of the images used in [22] and the ones we are interested for this study, it was decided to evaluate the feasibility of using this algorithm for wire detection.

3.2 Steger's Unbiased Detector of Curvilinear Structures

Next, the main ideas of the detection algorithm proposed by Steger are summarized for the 1D case. A more complete description, including its generalization to 2D, can be found in [22, 21, 23, 24].

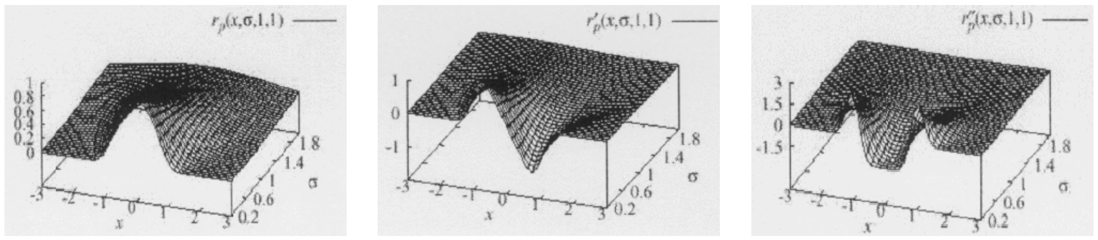
The algorithm is based on the concept that a line can be thought of as a one-dimensional manifold in $\mathcal{R}^2$ with a well defined width w. Similarly, curvilinear structures in 2D can be modeled as curves $s(t)$ that exhibit a 1D line profile in the direction perpendicular to the line – i.e. perpendicular to $s'(t)$.

Then, an ideal one-dimensional line profile can be modeled as a symmetrical bar-shaped profile given by

$$f_b(x) = \begin{cases} h, & \mid x \mid \leq w \\ 0, & \mid x \mid > w \end{cases} \tag{1}$$

where w is the width of the line and h is the contrast, or as a more general asymmetrical bar-shaped profiled given by

$$f_a(x) = \begin{cases} 0, & x < -w \\ 1, & \mid x \mid \leq w \\ a, & \mid x \mid > w \end{cases} \tag{2}$$



(a) (b) (c)

Figure 3: (a) Smoothed parabolic line profile. (b) Convolution with the first derivative of a Gaussian. (c) Convolution with the second derivative of a Gaussian.

where $a \in [0, 1]$. A more gradual drop between the line and the background can be modeled by a parabolic profile:

$$f_p(x) = \begin{cases} h(1 - (x/w)^2), & | x | \leq w \\ 0, & | x | > w \end{cases} \tag{3}$$

A line with a profile given by (3) can be found in an ideal noiseless image $z(x)$ by determining the points where $z'(x) = 0$. Salient lines can be identified by imposing that the magnitude of the second derivative $z''(x)$ at the point where $z'(x) = 0$ should be sufficiently large. In the presence of noise, as discussed in the previous section, the derivatives of the image should be estimated by convolving the image with the derivatives of a Gaussian smoothing kernel with standard deviation σ. The space-scale behavior of the smoothed parabolic profile and its convolution with the first and second derivative of Gaussian filters is shown in figure 3. It can be seen that it is possible to detect the *precise* location of the line *for all* σ. However, if the line follows a profile like (1) or (2), it can be shown that the magnitude of the convolution with the second derivative of the Gaussian has a clear maximum at the true image location *only* if

$$\sigma \geq \frac{w}{\sqrt{3}}$$

The width of the line can be estimated by looking at the places where the magnitude of the output of the first derivative of the Gaussian is maximum. It can be shown that if the line profile is symmetrical and the width is small the width will be estimated too large. Furthermore, in the case of parabolic profiles, the width will be estimated too large for a range of widths. In either case, the mapping between the estimated and the true width can be inverted and the true width can be determined with high accuracy. However, if the profile is not symmetrical, the estimated line location is biased towards the weak side of the line:

$$l = -\frac{\sigma^2}{2w} \ln(1-a)$$

but if a is known, the bias can be corrected.

3.3 Post Processing

After lines are detected using the above algorithm, noise and false alarms due to image clutter are reduced by rejecting short lines. This is accomplished by thresholding a Hough transform of the image obtained using the line pixels. The threshold used was fixed to

$$Th = \mathrm{mean} + 0.5(\max - \min) \tag{4}$$

where mean, $\max$, and $\min$ are the mean, maximum and minimum counter values in the parameter space, respectively.

4 Data Modeling and Simulation

In order to characterize the performance of the detection algorithm using statistical tests with a given accuracy, we must have large populations of sample representative images. Unfortunately, at the time of this study real testing data was unavailable. Therefore, realistic testing data was generated by combining real (background) images with synthetically generated wires that were corrupted using noise models. The procedure used to generate these images is described next.

4.1 Illumination Model

The scene illumination was assumed to have two distinct light sources: ambient light (i.e. diffused light from the landscape, sky and clouds) and a distant point light source (i.e. the Sun/Moon) as shown in figure 4. Furthermore, it was assumed that the ambient light impinged equally on all surfaces of the wire, from all directions. Thus, the reflected light from the wire to the image plane is given by

$$I = I_a K_a$$

where I_a is the intensity of the ambient light and K_a is the ambient reflection coefficient. For the point light source, the Phong illumination model was used. Thus, the reflected light from the wire to the image plane due to a point light source can be modeled as

$$I = f_{att} I_p [K_p cos(\theta) + w(\theta) cos^n(\alpha)]$$

where I_p is the light intensity of the point light source, f_{att} is the light attenuation factor[1], K_p is the diffusion-reflection coefficient of the wire surface w.r.t. the specific light spectrum of the light source, $w(\theta)$ is the material specular-reflection

[1]For example $f_{att} = min(\frac{1}{C_1 + C_2 r + C_3 r^2}, 1)$ or $f_{att} = \frac{1}{r^2}$, where r is the distance of the object from the light source.

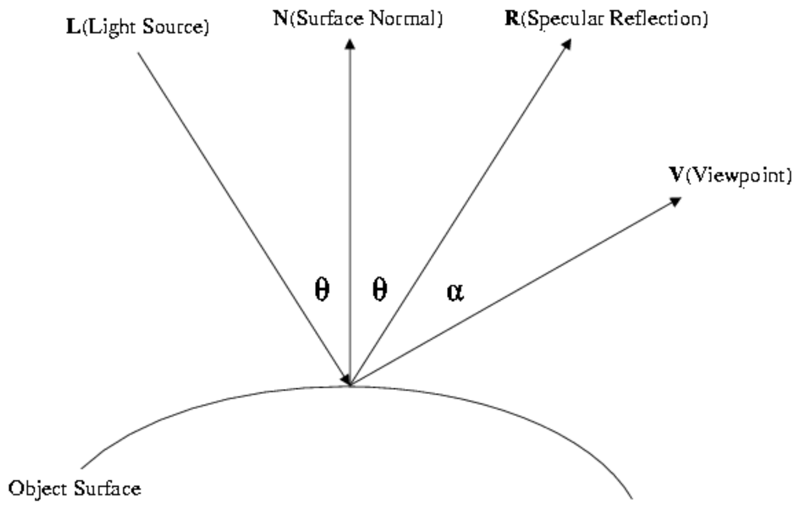


Figure 4: Illumination on a surface.

coefficient.

The typical thickness of power lines ranges between 5mm and 45mm. The typical cruising speed of a helicopter is between 100MPH and 400MPH. Thus, the distance from the camera on board the helicopter to the lines to be detected is such that the image width of the wire is typically less than 1 or 2 pixels. Therefore, the contribution from the specular-reflection component is insignificant and can be set to 0. In addition, the distance from the power lines to the point light source (the Sun) is very large and f_{att} can be set to 1.0. Thus, combining the ambient light model and the point light model, we have

$$I = I_a K_a + I_p K_p cos(\theta)$$

4.2 Coordinate Systems and Mapping Matrices

We will employ three coordinate systems: World System, Airborne System, and Camera System. The *world coordinate system* is a coordinate system fixed with the ground. Wire structures are stationary relative to this coordinate system. The *airborne coordinate system* moves with the helicopter and it has 6 degrees of freedom relative to the world coordinate system: 3 degrees of freedom for its origin (X_a, Y_a, Z_a) and 3 degrees of freedom for its orientation (α, β, γ), where α is

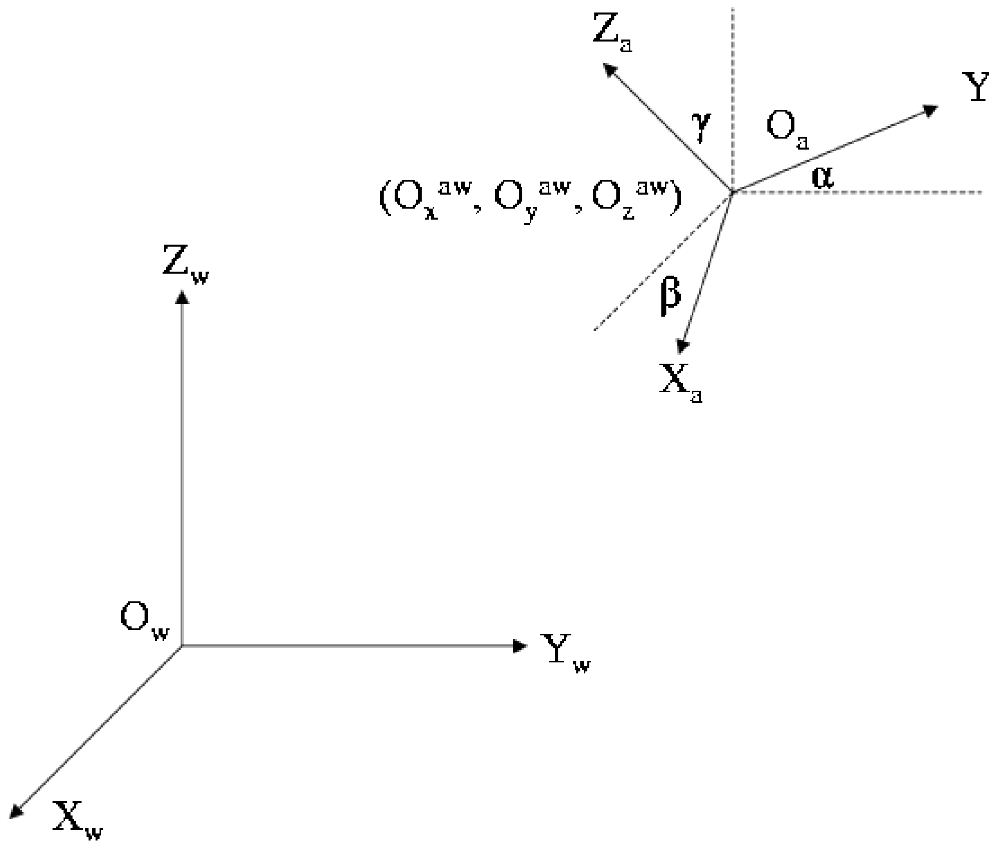


Figure 5: World and airborne coordinate systems.

called the angle of attack, β the yaw angle, and γ the roll angle (see figure 5). The transformation (mapping) matrix between the world and the airborne coordinate systems can be derived as:

$$\begin{bmatrix} X_a \\ Y_a \\ Z_a \end{bmatrix} = M_3 M_2 M_1 \left(\begin{bmatrix} X_w \\ Y_w \\ Z_w \end{bmatrix} - \begin{bmatrix} O_x^{aw} \\ O_y^{aw} \\ O_z^{aw} \end{bmatrix} \right) \tag{5}$$

where,

$$M_1 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\alpha & \sin\alpha \\ 0 & -\sin\alpha & \cos\alpha \end{bmatrix} \quad M_2 = \begin{bmatrix} \cos\beta & \sin\beta & 0 \\ -\sin\beta & \cos\beta & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad M_3 = \begin{bmatrix} \cos\gamma & 0 & -\sin\gamma \\ 0 & 1 & 0 \\ \sin\gamma & 0 & \cos\gamma \end{bmatrix} \tag{6}$$

The *camera coordinate system* is a 2D system fixed in the camera image plane. The pixel coordinates of the images captured by the camera are expressed in this coordinate system. The origin of the camera coordinate system is chosen to be the same as that of the airborne coordinate system (see figure 6). The mapping from the airborne system to the camera system is called the perspective transformation or imaging transformation. This transformation projects 3D points onto a 2D plane.

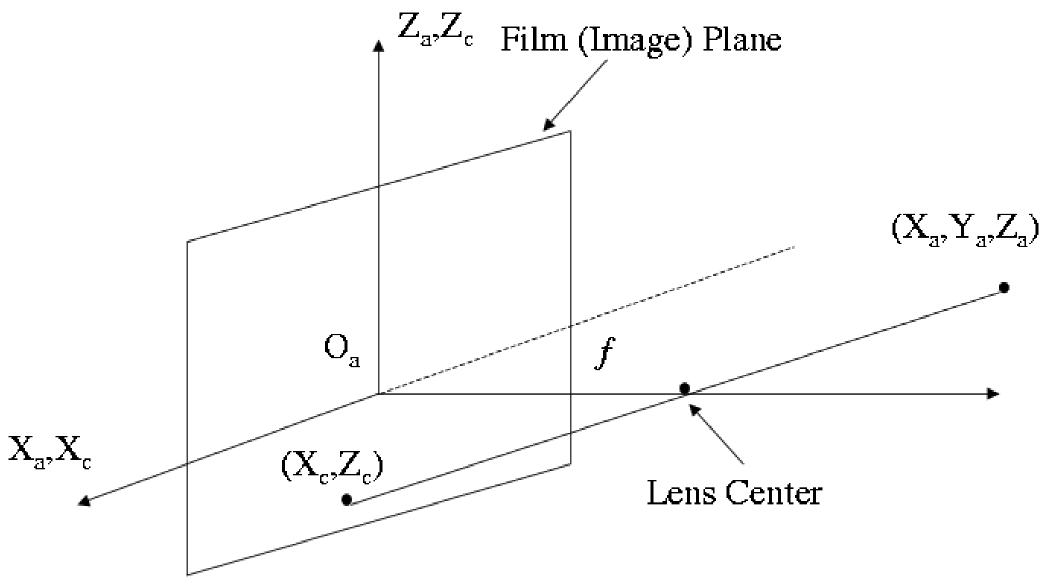


Figure 6: Camera and Airborne Coordinate Systems.

Unlike the regular coordinate transformation, the perspective transformation is a non-linear transformation. Let f be the focal length of the camera. Then, the mapping between the camera and the airborne coordinate systems is given by

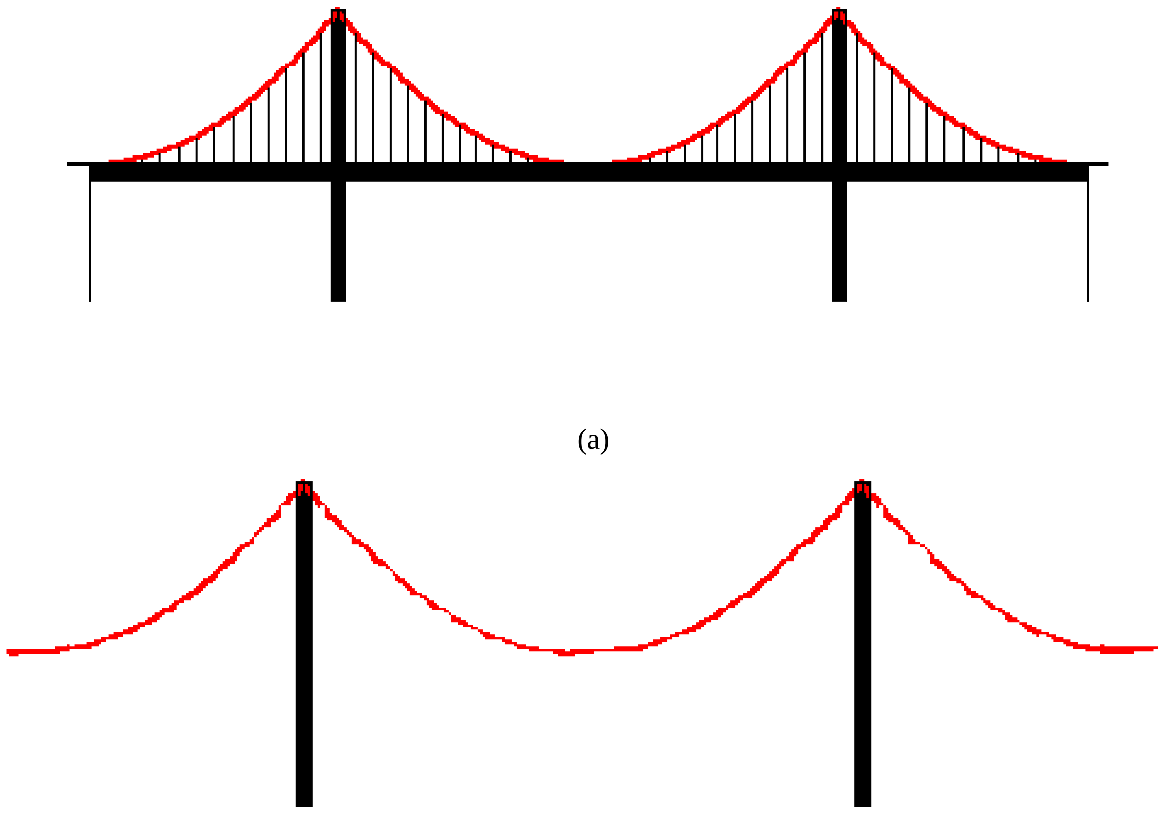
$$\begin{aligned} \frac{X_c}{f} &= -\frac{X_a}{Y_a - f} \\ \frac{Z_c}{f} &= -\frac{Z_a}{Y_a - f} \end{aligned} \tag{7}$$

4.3 Geometric Model

There are two kinds of curve structures which may be of importance to the present application: hanging bridges and power lines (see figure 7). Let μ be the linear weight density of the bridge deck, and T be the cable tension force. Then, with an appropriate choice of the coordinate system, the equation of the cables for the hanging bridge is that of a parabola,

$$y = \frac{\mu}{2T}x^2 \tag{8}$$

Similarly, let μ be the linear weight density of a power line, and T be the power line tension force. With an appropriate choice of the coordinate system, the equation of



(a)

(b)

Figure 7: (a) Cables of hanging bridges have a parabolic profile. (b) Cables from power lines have a catenary profile.

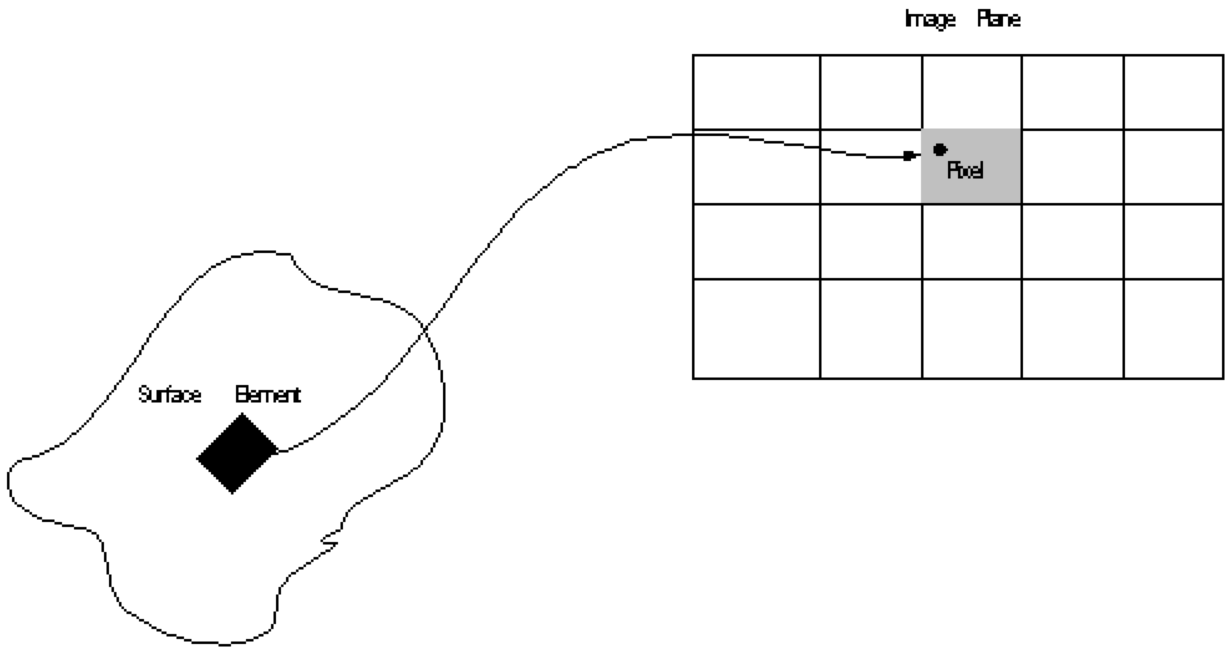


Figure 8: Mapping between wire structure element and an image pixel.

the power lines is the catenary curve:

$$y = \frac{T}{\mu} cosh(\frac{\mu}{T} x) - \frac{T}{\mu} \tag{9}$$

4.4 Image Generation of Wire Structures

Knowing the illumination and geometric models, we can use the mapping between the world and the camera coordinate systems to obtain synthetic images of wire structures. First, the surface of a wire structure is divided into finite elements. The size of each element is chosen so that its mapped area on the image plane is less than 1 pixel. Then, the reflected light from this element is mapped to the pixel within which the center of the element is mapped into (see figure 8). Finally, the pixel gray level value is computed as

$$I_{new} = I_{old} + A_e/A_{pixel} * (I_R - I_{old}) \qquad (10)$$

where I_{old} and I_{new} are the image value of this pixel before and after this surface element is mapped, respectively, I_R is the light intensity of the reflected light from this surface element, A_e is the mapped area of this pixel into the image plane[2], and A_{pixel} is the area of a pixel, which is a fixed size for a given digital camera. The image of a power line structure is generated when all surface elements have been mapped to the image plane.

4.5 Noise Model

Noise may be added to the generated image. The noise appears as breaks in the image of the wire structure. The location of the breaks is assumed to have a uniform distribution. The number of breaks follows a Poisson distribution, which has the

[2]Note that the mapped area may not be rectangular any more, even if the original element is.

following probability density function (pdf):

$$f_{poisson} = \frac{\mu^n}{n!} \exp^{-\mu} \tag{11}$$

where μ is the mean of the distribution. The size of the breaks (i.e. number of pixels for a break) is assumed to follow a Rayleigh distribution, whose pdf is

$$f_{rayleigh} = \frac{x}{\sigma^2} \exp^{-\frac{x^2}{2\sigma^2}}, \;\; x = 0, \ldots, +\inf \tag{12}$$

with mean $\frac{2}{\pi}\sigma$.

4.6 Background Image

For images captured from a low-flying helicopter, the background mainly consists of two things: clouds and landscape. Thus, to simulate realistic images, the background of the images were obtained by capturing real images using a digital camera. Then, computer generated images of power line structures were superimposed onto these real images.

5 Performance Evaluation of the Detection Algorithms

A performance evaluation protocol for the detection algorithms was designed based on the one described in [27]. The protocol measures to what extent the algorithm detects wires present in the image and whether the algorithm falsely detects wires in the background. These measurements are performed at both the pixel and the wire level.

5.1 Definitions and Notation

Before describing the protocol, the following terms must be defined:

Ground truth image (I_g). Original data, which is the basis of comparison with the detection result. Ground truth images are synthetically generated and consist of one or more dark wires on a white background. Each wire has a different pixel value which is used as an id of the wire.

Number of true wire pixels (P_g) . The number of ground truth image pixels that belong to a wire.

Detected image (I_d) Binary image, resulting after the scene (noisy) image has undergone the defined strategy for detecting edges (Steger's algorithm, followed by a Hough transform).

Number of detected wire pixels (P_d) . The number of detected image pixels that were labeled as belonging to a wire.

Overlap. If there is an edge pixel at position (x, y) in I_g and there exist an edge pixel at the same position (x, y) or any of its eight neighbors in I_d, then an overlap is said to occur between the two edge pixels. Let $P_{g \times d}$ be the number of overlap pixels between I_g and I_d.

5.2 Pixel Level Performance Indices

Next we define a set of indices to measure the performance of the algorithm at the *pixel level*. These indices provide information about the number of wire pixels correctly and incorrectly labeled.

True Positives or Pixel Detection Rate (PDR). The *pixel detection rate* (PDR) is the rate of positive responses in the presence of instances of the sought feature:

$$PDR = \frac{P_{g \times d}}{P_g} \tag{13}$$

Pixel False Positives or False Alarms (PFA). The *pixel false alarm* (PFA) is the rate of positive responses in the absence of the sought feature:

$$PFA = 1 - \frac{P_{g \times d}}{P_d} \tag{14}$$

Pixel Recovery Index (PRI). The *pixel recovery index* (PRI) combines the PDR and the PFA in a single index:

$$PRI = \alpha PDR + (1-\alpha)PFA, \ \ 0 \leq \alpha \leq 1 \tag{15}$$

where α weights the relative importance of true positives and false alarms. (In our study, $\alpha = 0.5$.)

5.3 Wire Level Performance Indices

The pixel level performance criteria defined above do not provide a measure of how many individual wires or which fragments of each wire were detected. For this purpose, the following *wire level indices* are defined:

Wire Detection Rate (WDR). A wire is said to be detected if a number greater than a threshold (in our case 50%) of its pixels are detected. The *wire detection rate* (WDR) is the ratio of the total number of wires detected to the total number of wires in the ground truth image.

$$WDR = \frac{\text{Number of wires in } I_d}{\text{Number of wires in } I_g} \tag{16}$$

Detection Fragmentation Rate (DFR) A measure of the fragment of each wire detected. The *detection fragmentation rate* (DFR) is defined as

$$DFR = \frac{\text{Number of pixels detected in a wire}}{\text{Number of pixels in the wire}} \tag{17}$$

6 Experimental Results

Synthetic images were generated following the procedure described in section 4. Each image had three wires of different diameters (18 mm., 21.5 mm. and 45 mm.) viewed from different distances ranging between 560 m. to 2,800 m. Figure 9(a) and (b) show images where the time to collision is 25 seconds for helicopter speeds of 100 km/h (694.4 m), and 400 km/h (2777.78m), respectively. Edges were detected in the synthetic images by using an implementation of Steger's algorithm provided by the author. Examples of the results are shown in figure 9(c) and (d). The results of post processing using the Hough transform are shown in figure 9(e) and (f).

The three pixel level indices for the different cable distances are shown in figure 10. As expected, the performance degrades as the distance increases. Due to time constraints, the results illustrated here were obtained by applying only Steger's algorithm, without post processing. While the false alarms are relatively high, as it is seen in figure 9 (e) and (f) post processing does eliminate most of the false alarms.

The wire detection rate and the detection fragment rate are shown in figures 11 and 12, respectively. These plots show that most of the misdetection errors are due to the thinnest of the wires, indicating a limitation on the diameter of the wires that can be safely detected.

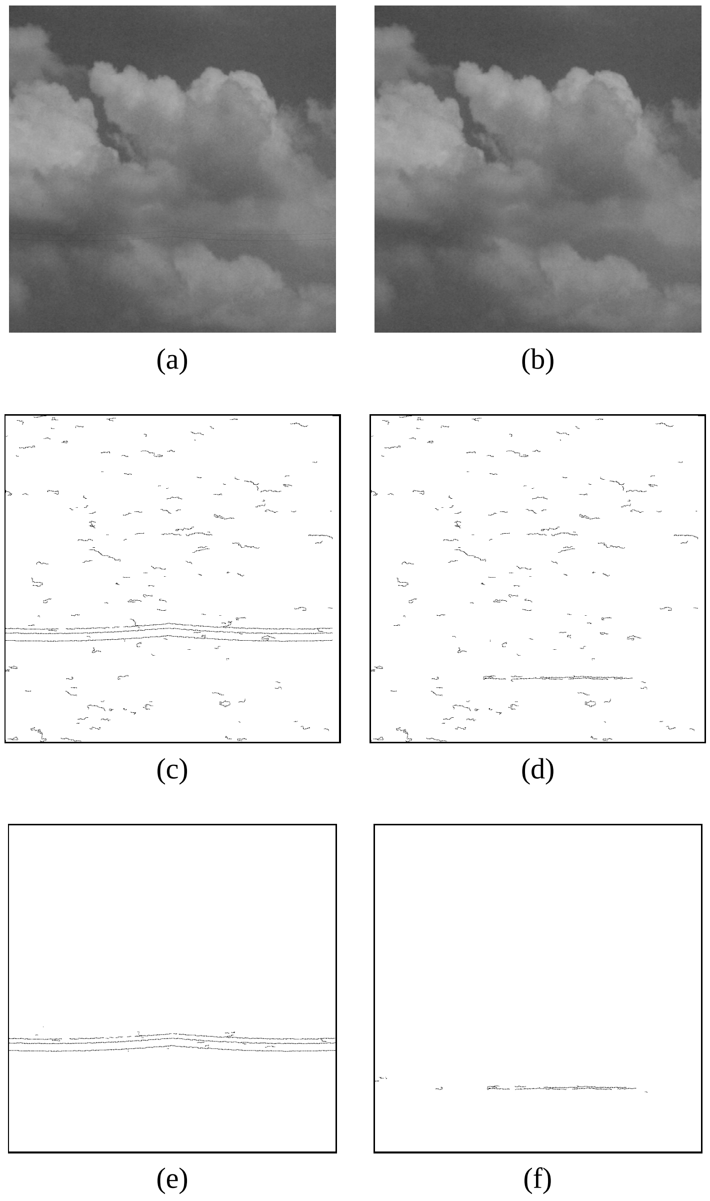


Figure 9: Three cables (ϕ = 18 mm, 21.5 mm, 45 mm) at 25 seconds to collision for a helicopter moving at (a) 100 km/h and (b) 400 km/h; (c) and (d) edges detected in (a) and (b) using Steger's algorithm; (e) and (f) edges after Hough transform of (c) and (d).

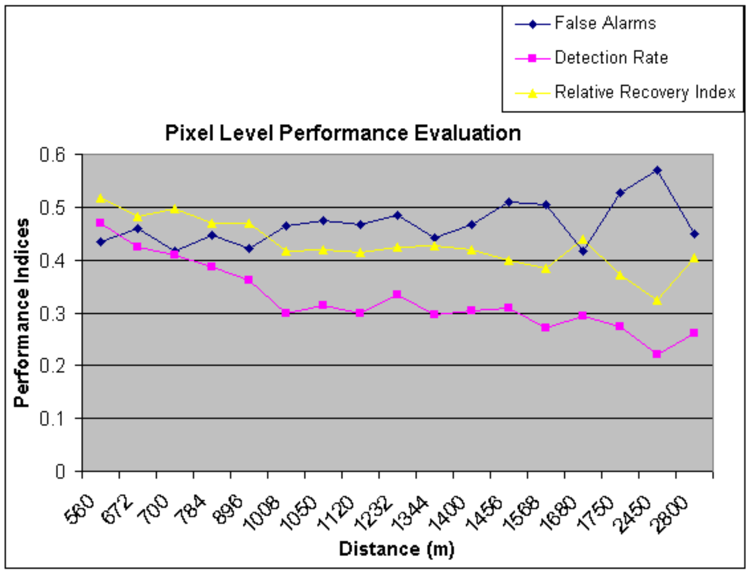


Figure 10: Pixel level indices (PDR, PFA, PRI) vs distance.

7 Support Vector Machines

In this section we briefly describe Support Vector Machines and outline some of the advantages. Support Vector Machines(SVMs) have been proposed as an effective pattern recognition tool in the recent past. SVMs have been used in various applications such as text categorization [28, 13], face detection [19] and so on. They can cope with high dimensional data quite easily and have good generalization capability. Generalization capability refers to the ability to perform well on unseen test data.

The first property that distinguishes the SVM from other nonparametric techniques

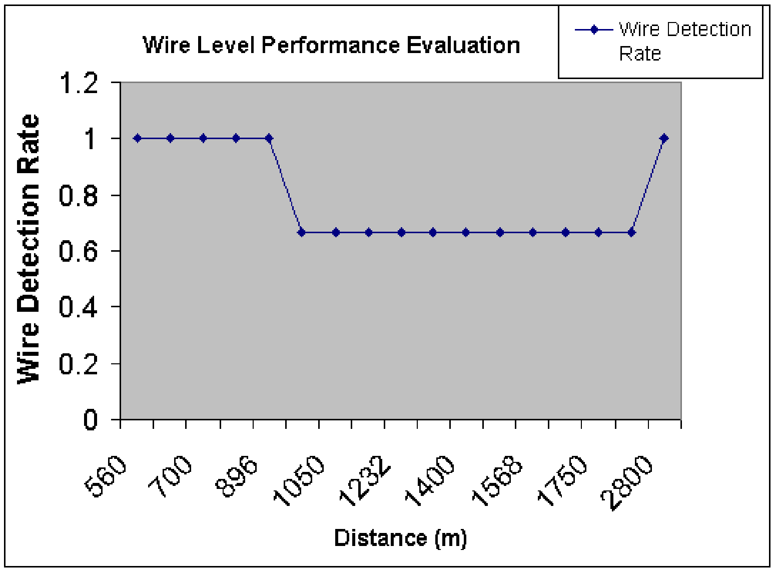


Figure 11: Wire level performance evaluation.

such as neural networks is that SVMs minimize the structural risk i.e., the probability of misclassifying a previously unseen data point drawn randomly from a fixed but unknown probability distribution instead of minimizing the empirical risk i.e., the misclassification on training data. Thus SVMs have good generalization. For a given learning task, with a given finite amount of training data, the best generalization performance will be achieved if the right balance is struck between the accuracy on that training set and the *capacity* of the machine to learn any training set without error [1]. Thus capacity control is important for good generalization. SVMs allow for good generalization capability since the number of parameters re-

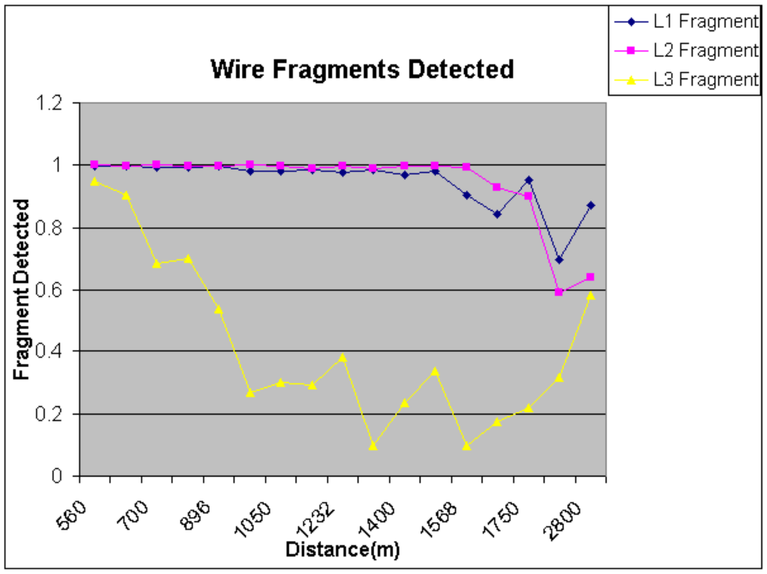


Figure 12: Wire level performance evaluation (fragments detected).

quired for "capacity control" is very small. Hence they provide a trade-off between accuracy and generalization capability. The concept of capacity is not discussed here. The interested reader is referred to Vapnik [26] and Burges [1] for a detailed discussion on capacity.

Secondly, an SVM condenses all the information contained in the training set relevant to classification in the support vectors. The support vectors are a subset of the training set. These are the only vectors required for defining the decision surface and hence the only ones relevant for classification. This effectively reduces the size of the training set, identifying the most important points, and makes it possible to efficiently perform classification. Finally SVMs are quite naturally designed to perform classification in high dimensional spaces, especially where the data is generally lineraly non-separable. SVMs can be used to construct models which are simple enough to analyze mathematically yet complex enough for real world applications.

The training of an SVM is actually the solution of a Quadratic Programming Problem. (see Appendix A and B). The Quadratic Problem is usually dense and quite large. However a number of techniques have been developed to deal with large quadratic programming(QP) problems and have hence made SVMs more accessible. Some of the strategies include "chunking" [25], decomposition into sub-problems [19] and a number of Active Set methods. Sequential Minimal Optimization(SMO) [20] is an algorithm that can be used for fast training of SVMs. Joachims [12] describes an algorithm for large SVMs. The software which runs on this algorithm is called SVM^{Light}. Both SMO and SVM^{Light} can be classified as active set methods. In this report SVM^{Light} was used for implementation.

8 Experiments with Support Vector Machines

In this section we describe in detail, the experiments performed using Support Vector Machines. The procedure employed to select training data and other implementation details are discussed in the subsequent subsections. Preliminary results are shown in section 9.

8.1 Training the SVM

Real images were used for these experiments. Masks extracted from an image were used as training data for the SVM. The procedure for the extraction of masks is described in section 8.1.1. The image used is shown in Figure 13. It is of size 240 $\times$ 180. The training set consisted of positive and negative examples. The training set consisted of 400 dimensional vectors(20 $\times$ 20 masks) and consisted of positive and negative examples. The selection of training data is explained in Section 8.1.2. SVM^{Light} [12] software was used both for training and classification. This software is free for scientific use and the source code may be downloaded from

ftp://ftp-ai.cs.uni-dortmund.de/pub/Users/thorsten/svm_light/current/svm_light.tar.gz

Linear, polynomial and Gaussian Radial Basis Function(RBF) kernels were tried.Only the Gaussian Radial Basis Function(RBF) kernel was found to be useful for these experiments. (see Appendix B for an overview of kernels and refer table 3 for a listing of commonly used kernels.) The equation used may be stated as:

$$K(\vec{x}, \vec{y}) = exp(-\gamma \mid\mid (\vec{x} - \vec{y}) \mid\mid^2)$$

The parameters used for the experiments were:

- C(Penalty term) = 0.1,1,10

- γ = 1e-4

The results corresponding to C = 1 and γ = 1e-4 are shown.

Two sets of training data were used in training the SVM. These will be referred to as **Training Set 1** and **Training Set 2** respectively. The results corresponding to both these data sets is shown. These training sets are described in detail subsequently.

8.1.1 Procedure for extracting masks from the image

A mask of size 20×20 was run over the entire given image. The displacement between 2 successive masks is 2 pixels (either in the X or in the Y direction i.e.along columns or along rows). Each of these masks is used as a test case for classification, in order to determine whether or not there is a wire at that location. Thus the test vectors are 400 dimensional vectors. The total number of masks extracted from this image by the above procedure was 8800, which means that there are potentially 8800 possible locations in the image where there could be a wire.

8.1.2 Training Data Selection

Consider a mask of size 20×20. The mask was divided into three bands which we shall refer to as **Top**, **Middle** and **Bottom** bands respectively. They were as below. For the following discussion, indexing is taken to begin from 1. Thus the mask consists of rows 1-20 of pixels and columns 1-20 of pixels.

1. **Top** : First 7 rows of pixels. (Rows 1-7)
2. **Middle** : Middle 6 rows of pixels. (Rows 8-13)
3. **Bottom** : Last 7 rows of pixels. (Rows 14-20)

Of the masks extracted, those which contained wires strictly within the middle band i.e. those in which the wire was contained entirely within rows 8-13, were selected as positive examples. Among these, some of them were used as positive training examples and some others were used as positive test cases for cross validation. The following types of masks comprised the set of negative examples.

1. Containing only background and no traces of wire
2. Containing background and only “vestiges” of a wire
3. Containing a wire entirely within top band

Example type(+ve or -ve)	Description	Training	Cross validation
Positive examples	Wire strictly within the middle band	400	72
Negative examples	Background only	360	140
	Background + “vestiges” of wire	20	20
	Wire in top band	180	60
	Wire in bottom band	40	20

Table 1: Breakdown of training and cross validation sets(Training Set 1)

	Positive	Negative	Total
Training	400	600	1000
Cross validation	72	200	272

Table 2: Summary of training and cross validation sets(Training Set 1)

4. Containing a wire entirely within bottom band

Among the numerous negative examples, a total of 800 was selected for training and cross validation. Again some of the 800 were selected as negative training examples and others as negative test cases. Masks which were hard to categorize as belonging to any of the three bands(e.g. those which overlap top and middle bands or top and bottom bands) were regarded as “**Transition Band Masks**” and were not considered either for training as well as cross validation. The breakdown of the training and cross validation sets is given in Table 1. The overall numbers for the training and test sets are summarized in Table 2.

9 Preliminary results of experiments with Support Vector Machines

In this section we present preliminary results of experiments performed using Support Vector Machines.

9.1 Results for training set 1

The original image used for training as well as cross validation and the results of classification are shown in Figure 13. The results on the cross validation set are as below:

Number of support vectors = 435

Number of missed detections = 0

Number of false alrams = 0

Precision/Recall on cross validation set = 100%/100%

Figure 14 shows the detected locations superimposed on the original image.

* The black pixels correspond to the locations where a wire has been detected. A mask is run over the image and each mask is classified as to whether it contains a wire or not. The locations of the black pixels in the result image are actually the centre of those masks in which a wire has been detected.

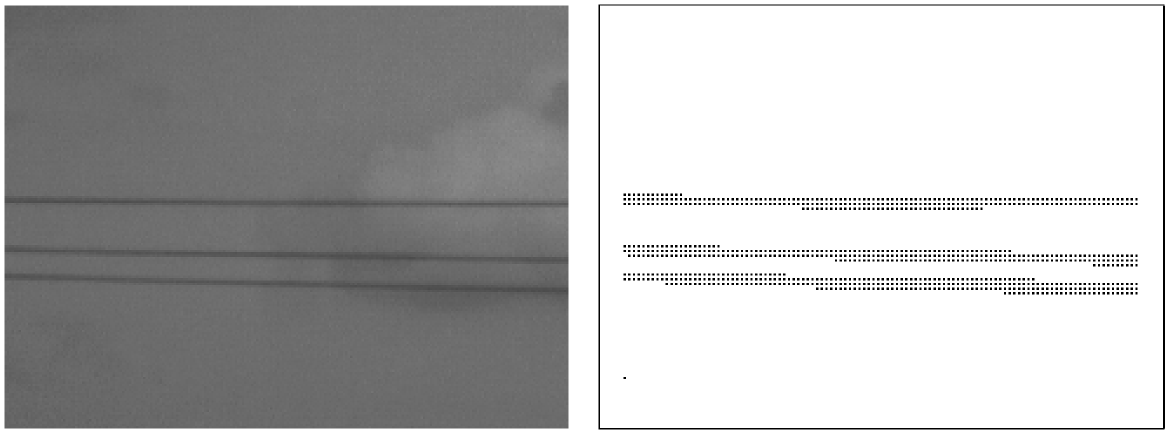

(a) (b)

Figure 13: (a) Original image used for testing. (b) Results of classification*

9.2 Results for training set 2

The experiment was repeated for the same image and a different training and cross validation set. The new set was almost the same as the previous set (refer Table 1) except for the following differences.

1. Number of positive training examples = 200
2. Number of positive test examples(cross validation) = 473

Thus the total number of training examples was

200(positive) + 600(negative) = 800

and the total number of test examples(for cross validation) was

473(positive)+200(negative) = 673.

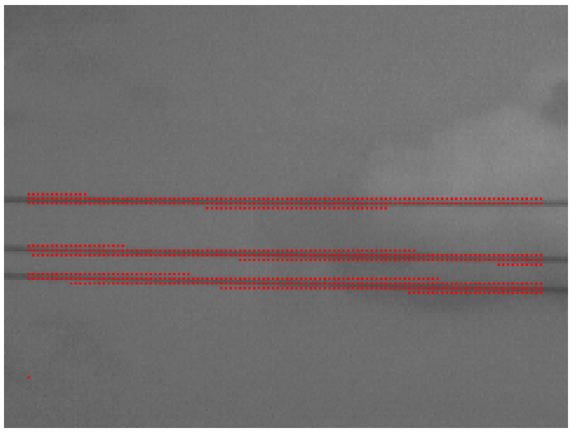


Figure 14: Detected locations superimposed on the original image(Training Set 1)

The parameters used for the SVM were the same as previous case(Equation A.3). i.e. Gaussian Radial Basis Function Kernel, C = 0.1,1,10,100 $\gamma = 1e - 4$. The results for C = 1 are outlined below. The detected locations of the wire and the original image are shown in Figure 15

Number of support vectors = 332

Number of missed detections = 37

Number of false alarms = 0

Precision/recall on test set: 100.00%/92.18%

Results for three other images are shown in the figures 16, 17 and 18. A parameter(p) was varied to control the amount of false alarms. This parameter has a value between 0 and 1, 0 corresponding to no control of false alarms. Results are shown

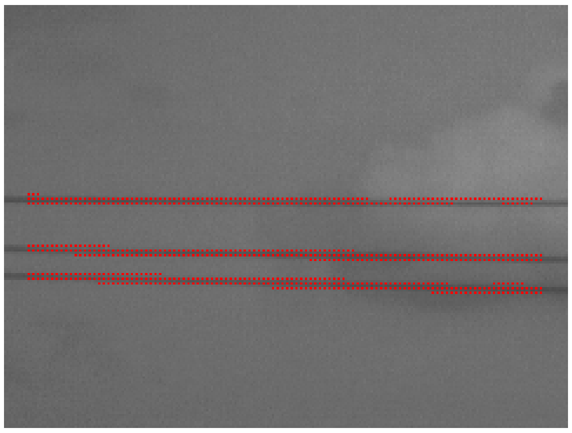


Figure 15: Detected locations superimposed on the original image(Training Set 2)

for images for p = 0.3.

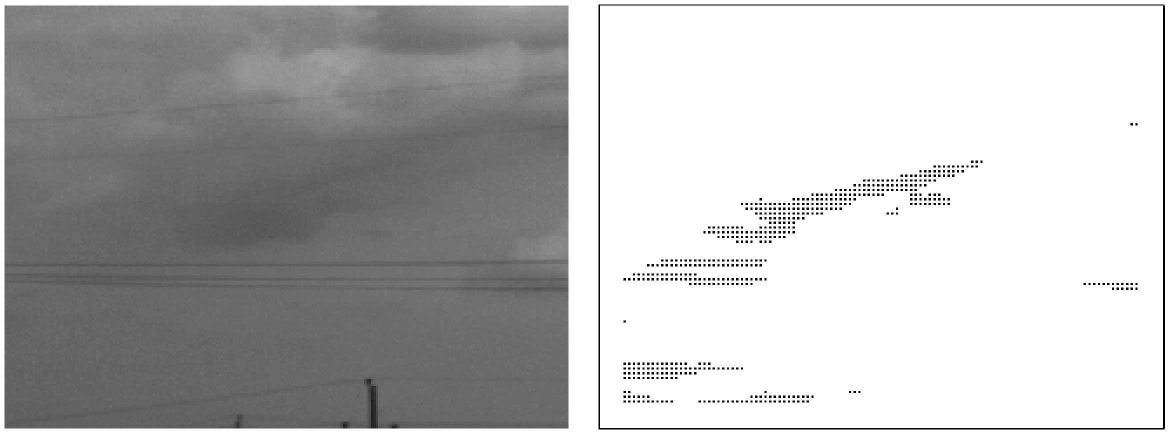

(a) (b)

Figure 16: (a) Image 2.(Original image) (b) Detections for p=0.3

10 Summary and Conclusions

In this report we addressed the problem of obstacle detection for low-altitude rotorcraft navigation with emphasis on wire detection. A line detector with sub pixel accuracy proposed by Steger was identified from the published literature. Steger's algorithm was tested using a set of synthetically generated images combining real backgrounds with computer generated wire images. A set of performance indices at the pixel and the wire level were defined to evaluate the merits of the algorithm for the task at hand. The results of the experiments show that the algorithm can potentially detect wires, provided that they are not too thin or very far. It was also

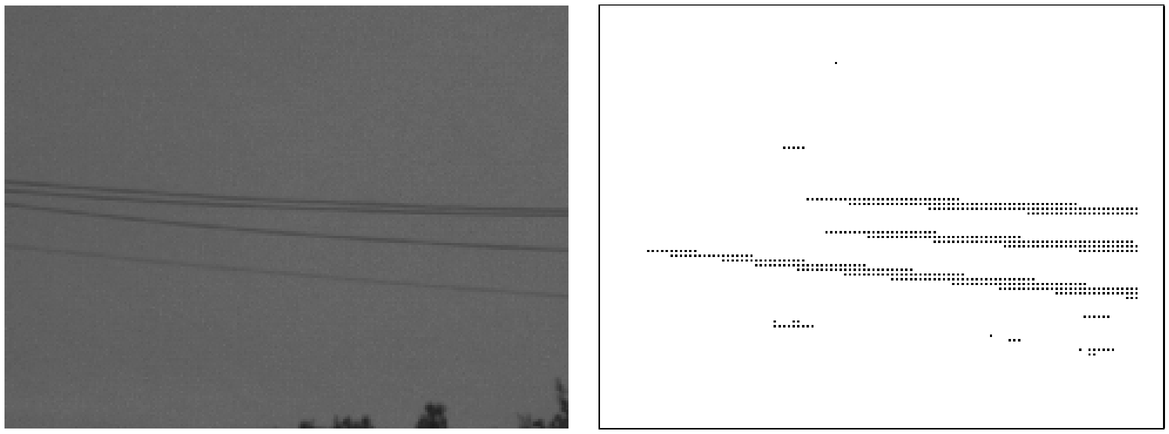

(a) (b)

Figure 17: (a) Image 3.(Original image) (b) Detections for p=0.3

observed that the algorithm produces false alarms due to the severe image clutter. However, most of these false alarms can be successfully eliminated by using a simple – albeit time consuming – post processing such as a Hough transform that discards short lines.

Future research should explore 1) integration over time of the obtained results to detect very thin (or distant) wires and 2) use image context – i.e. search for wires near power poles.

The use of an example-based learning technique, namely Support Vector Machines (SVMs) was also explored. Support Vector Machines have emerged as a promising classification tool and have been used in various applications. It was found that this approach is not very suitable for very thin wires and of course, not suitable

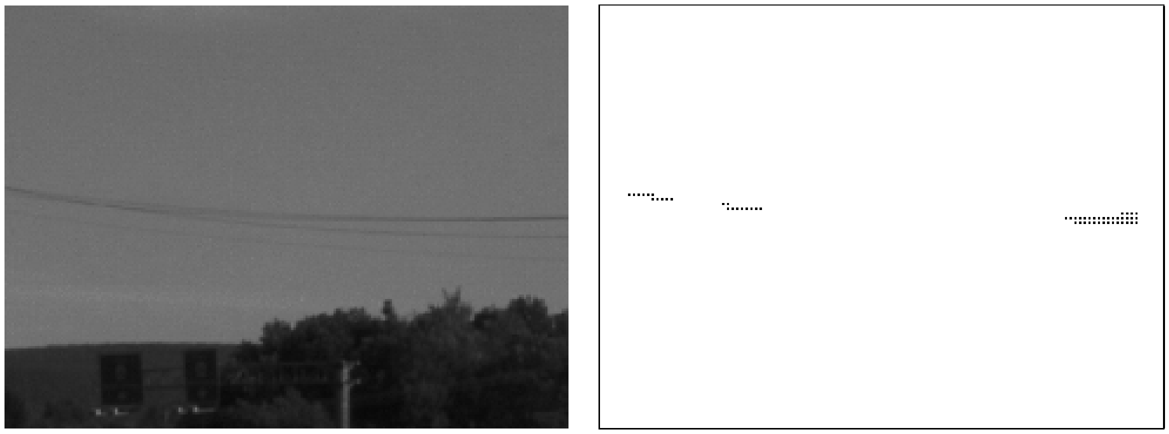

(a) (b)

Figure 18: (a) Image 4.(Original image) (b) Detections for p=0.3

at all for sub-pixel thick wires. High dimensionality of the data as such does not present a major problem for Support Vector Machines since they cope with high dimensional data fairly easily. However it is desirable to have a large number of training examples especially for high dimensional data. The main difficulty in using SVMs (or any other example-based learning method) is the need for a very good set of positive and negative examples since the performance depends on the quality of the training set. A large number of carefully chosen “good” negative examples is required in order for the classifier to learn the class corresponding to the negative examples (in our case non-wire). By “good” negative examples we mean those which could be easily confused with positive examples.

References

[1] C. Burges. A tutorial on support vector machines for pattern recognition. *Data Mining and Knowledge Discovery*, 2:121–167, 1998.

[2] A. Busch. A common framework for the extraction of lines and edges. *Int. Archives of Photogrammetry and Remote Sensing*, 31 part B3:88–93, 1996.

[3] G. Chen and Y. H. H. Yang. Edge detection by regularized cubic b-spline fitting. *T-SMC*, 25:636–643, 1995.

[4] D. Eberly, R. Gardner, B. Morse, S. Pizer, and C. Scharlach. Rigdes for image analysis. Technical Report TR93-055, Dept. of Computer Science, Univ. of North Carolina, Chapel Hill, 1993.

[5] M. A. Fischler, J. M. Tenenbaum, and H. C. Wolf. Detection of roads and linear structures in low-resolution aerial imagery using a multisource knowledge integration technique. *Computer Graphics and Image Processing*, 15:201–223, 1981.

[6] T. Gandhi. *Image Sequence Analysis for Object Detection and Segmentation*. PhD thesis, The Pennsylvania State University, Dept. of Computer Science and Engineering, December 1999.

[7] D. Geman and B. Jedynak. An active testing model for tracking roads in satellite images. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 18(1):1–14, 1996.

[8] A. Goshtasby and H. L. Shyu. Edge detection by curve fitting. *IVC*, 13:169–177, 1995.

[9] R. M. Haralick, L. T. Watson, and T. J. Laffey. The topographic primal sketch. *Int. Robotics Research*, 2(1):50–72, 1983.

[10] Marti A. Hearst. Support vector machines. *IEEE Intelligent Systems*, pages 18–21, June/July 1998.

[11] R. Jain, R. Kasturi, and B G. Schunck. *Machine Vision*. McGraw-Hill Inc., 1995.

[12] T. Joachims. Making large-scale svm learning practical. In B. Schlkopf, C. Burges, and A. Smola, editors, *Advances in Kernel Methods - Support Vector Learning*. MIT-Press, 1999. Software downloadable from

ftp://ftp-ai.cs.uni-dortmund.de/pub/Users/thorsten/svm_light/current/svm_light.tar.gz.

[13] Thorsten Joachims. Text categorization with support vector machines: learning with many relevant features. In Claire Nédellec and Céline Rouveirol, editors, *Proceedings of ECML-98, 10th European Conference on Machine Learning*, number 1398, pages 137–142, Chemnitz, DE, 1998. Springer Verlag, Heidelberg, DE.

[14] R. Kasturi, O. Camps, L. Coraor, T. Gandhi, and S. Devadiga. Detection of obstacles on runway using ego-motion compensation and tracking of significant features. Technical report, Dept. of Computer Science and Engineering, The Pennsylvania State University, 1996.

[15] R. Kasturi, O. Camps, L. Coraor, T. Gandhi, K. Hartman, and M. T. Yang. Obstacle detection algorithms for aircaft navigation. Technical report, Dept. of Computer Science and Engineering, The Pennsylvania State University, January 2000.

[16] I. S. Kweon and T. Kanade. Extracting topographic terrain features from elevation maps. *Computer Vision, Graphics, and Image Processing: Image Understanding*, 59(2):171–182, 1994.

[17] J. B. A. Maintz, P. A. van den Elsen, and M. A. Viergever. Evaluation of ridge seeking operators for multimodality medical image matching. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 18(4):353–365, 1996.

[18] Jorge Nocedal and Stephen Wright. *Numerical Optimization*. Springer-Verlag New York Inc., 1999.

[19] Edgar E. Osuna, Robert Freund, and Federico Giorsi. Support vector machines:training and applications. Technical report, MIT Artificial Intelligence Laboratory and Center for Biological and Computational Learning Department of Brain and Cognitive Sciences, March 1997.

[20] John C. Platt. Fast training of svms using sequential minimal optimization. In B. Schlkopf, C. Burges, and A. Smola, editors, *Advances in Kernel Methods - Support Vector Learning*, pages 185–208. MIT Press, 1998.

[21] C. Steger. Analytical and empirical performance evaluation of subpixel line and edge detection. In Kevin J. Bowyer and P. Jonathon Phillips, editors, *Empirical Evaluation Methods in Computer Vision*, pages 188–210, Los Alamitos, California, 1998. IEEE Computer Society Press.

[22] C. Steger. An unbiased detector of curvilinear structures. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(2):113–125, Feb 1998.

[23] C. Steger. Subpixel-precise extraction of watersheds. In *7th International Conference on Computer Vision*, volume II, pages 884–890, 1999.

[24] C. Steger. Subpixel-precise extraction of lines and edges. In *International Archives of Photogrammetry and Remote Sensing*, volume XXXIII, part B3, pages 141–156, 2000.

[25] V. Vapnik. *Estimation of Dependencies Based on Empirical Data*. Springer-Verlag, 1982.

[26] V.N. Vapnik. *The Nature of Statistical Learning Theory*. Springer-Verlag, 1995.

[27] L. Wenyin and D. Dori. A protocol for performance evaluation of line detection algorithm. *Machine Vision Applications*, 9(5/6):240–250, 1997.

[28] Y. Yang. An evaluation of statistical approaches to text categorization. *Journal on Information Retrieval*, 1998.

A Appendix: Overview of SVM theory

Problem Statement

The statistical pattern recognition problem is stated formally as below.

To estimate a function $f : R^N \rightarrow \{\pm 1\}$ using training data i.e. $(x_1, y_1), \ldots, (x_l, y_l) \in R^N \times \{\pm 1\}$ where x_i s are N dimensional vectors and y_i s are class labels such that f will correctly classify new examples (x, y) which were generated from the same underlying probability distribution $P(x, y)$ as the training data. [10]

It is important to understand that minimizing training error does not imply a small expected test error. Moreover it is crucial to limit the class of learning functions the classifier can implement to one with a *capacity* suitable for the amount of training data available [10]. The interested reader is referred to Vapnik [26] and Burges [1] for a detailed discussion on **capacity**.

A.1 Terms and concepts

We restrict our attention to the two class case for the rest of the discussion. A few terms which will be referred to frequently in the succeeding discussion are explained below.

- **Hyperplane** : Generalization of a line in higher dimensions. The equation of a hyperplane may be specified as

$$\vec{w}.\vec{x} + b = 0 \tag{18}$$

  where $\vec{w}$ and b are constants and $x \in R^N$

- **Margin** : Minimum separation between any data point and the separating hyperplane.

- **Linearly separable data** : If a hyperplane can be found such that all the data points that belong to one class are on one side of the hyperplane and all the data points of the other class are on the other side of the hyperplane, the data is said to be linearly separable.

The following are specifically related to SVMs.

1. **Optimal Hyperplane**

   The optimal hyperplane is defined as the one with the maximal margin of separation between the two classes.(This has the lowest *capacity* for the given training set.)

2. **Support Vectors**

   The training vectors which lie on the margin.

   (specifically at a distance equal to the margin from the separating hyperplane.)

Significance of support vectors

1. Only they are required to define the separating hyperplane. i.e. The separating hyperplane may be expressed as a combination of these vectors alone. (more generally, as a linear combination of functions of these vectors.)
2. Other examples may be ignored/moved around as long as they do not cross over to the other side.

A.2 Support Vector Machines:Description

In this subsection we describe the technique of Support Vector Machines(SVM). Given a set of points that belong to one of the two classes (training data), the SVM explicitly attempts to maximize the margin, i.e. it finds the hyperplane with the maximum margin. This hyperplane minimizes the risk of misclassifying (unseen) examples of the test set, provided the training and test data are drawn from the same (fixed but unknown) probability distribution [26].

The SVM minimizes the structural risk [25] and this results in better generalization capability. This is in contrast to some of the traditional pattern recognition techniques such as neural networks which minimize the empirical risk i.e. they minimize the error on the training set. This could result in poor performance on a test set (previously unseen examples) even though the performance on training set is very good, a phenomenon referred to as *overfitting*.

The SVM problem formulation leads to a Quadratic Programming (QP) problem. Although the QP is dense, it is convex. (see Appendix B). A convex QP has a global minimum and a number of nice properties which can be exploited by algorithms in solving it. There are a number of methods in the literature for convex QPs. An interesting class of methods is the Active Set methods. Although most active set methods can be generalized to solve non-convex QPs, this case presents significant complications. Nocedal and Wright [18] describe QP in detail. The interested reader is referred to [18, Chapter 16] for a detailed discussion on Active Set methods.

We consider three cases as in Osuna et. al. [19, Section 2.3]. These are illustrated in Figure 19.

We note that in the case of linearly separable data there is no unique hyperplane that separates the data. (Similarly in the case of data separable by a non-linear decision surface, there is no unique surface that separates the data.) This is illustrated in Figure 19(a).

The SVM finds the hyperplane that maximizes the margin. Figure 19(a) intuitively illustrates the importance of a large margin. The margin may be viewed as a measure of tolerance to noise. Note that some of the data points are much closer to the alternate hyperplane as compared to the one determined by the SVM. Hence, the "magnitude" of the noise required to "drive away" these data points to the other side of the alternate hyperplane (and hence leading to these points being misclassified) is much lesser as compared to the hyperplane determined by the SVM. Thus, intuitively we would expect the SVM to be more robust.

Each of the three cases is briefly described as follows.

1. **Linear Classifier and Linearly separable problem**

   This case (see Figure 19(a)) corresponds to finding a hyperplane $(\vec{w}, b)$ assuming that the data is linearly separable. We construct and solve the dual problem (described in section B.1.1) since the dual problem can be easily generalized to include linearly non-separable data.

2. **Linearly Non-Separable Case : Soft Margin Hyperplane**

The soft margin hyperplane is a generalization of the first case. This applies to the scenario in which the data is linearly separable except for a small fraction of points (see Figure 19(b)). The aim still is to find a hyperplane which separates the data. The main difference is that there is a penalty incurred for every training example which "violates" the separation by the hyperplane. In essence, the aim is to find a separating hyperplane which incurs the least penalty overall.

Making the penalty linearly related to the distance from the separating hyperplane for the "violating" examples somewhat simplifies the mathematics. In this case the penalty is expressed in the form of a penalty term C, which is the penalty per unit distance from the separating hyperplane (on the wrong side of it). This leads to the problem formulation as described in section B.2.

3. **Nonlinear Decision Surfaces**

   The SVM problem formulation can be easily generalized to include nonlinear decision surfaces (see Figure 19(c)) with the aid of **kernels**. The basic idea is to project all the data from the input space to a high dimensional space in which the data are linearly separable. With the aid of *kernels* (and considering the dual problem) the SVM problem formulation for this case is almost identical to that for the linearly separable case. A significant advantage with kernels is that all computations can be performed in the input space itself rather than in the high dimensional space. Since kernels are important in a discussion on SVMs and also are widely used in real world applications, an overview is presented here.

A.3 Kernels

The basic idea behind non linear decision surfaces may be outlined as below.

1. Map the input space into a (higher dimensional) feature space such that the data is linearly separable in the feature space. Specifically, an input vector $\mathbf{x}$ is mapped into a (possibly infinite) vector of "feature" variables as below:

   $$\mathbf{x} \rightarrow \phi(\mathbf{x}) = (a_1\phi_1(\mathbf{x}), a_2\phi_2(\mathbf{x}), \ldots, a_n\phi_n(\mathbf{x}), \ldots) \tag{19}$$

   where $\{a_i\}_{i=1}^{\infty}$ are real numbers and $\{\phi_i\}_{i=1}^{\infty}$ are real functions.

2. Find the *optimal* hyperplane in the feature space. The optimal hyperplane is defined as the one with the maximal margin of separation between the two classes. (This has the lowest *capacity* for the given training set.)

We discuss kernels from the point of view of SVMs. Consider any two training vectors $\vec{x}$ and $\vec{y}$. It is to be observed that in all problem formulations and the final decision function, the support vectors enter into the expressions only in the form of dot products.(see equations 29 and 33). Equation (33) is repeated here for reference.

(Equation 33)

$$f(\mathbf{x}) = sign(\phi(\mathbf{x}).\mathbf{w}^* + b^*) \Rightarrow sign(\sum_{i=1}^{l} y_i \lambda_i^* \phi(\mathbf{x}).\phi(\mathbf{x_i}) + b^*)$$

It is thus convenient to introduce a *kernel* function K such that :

$$K(\vec{x}, \vec{y}) = \phi(\vec{x}).\phi(\vec{y}) \tag{20}$$

Using this quantity the decision function of the SVM is :

$$f(\vec{x}) = sign(\sum_{i=1}^{1} y_i \lambda_i^* K(\vec{x}, \vec{x_i}) + b)$$

where

$\vec{x}$	=	The test vector which must be classified as belonging to class 1 or -1
l	=	Number of support vectors
x_i	=	ith support vector
λ_i^*	=	Lagrange parameter of the i th support vector.
b	=	bias

The advantages of using kernels may be outlined as follows.

- The mapping function $\phi(.)$ need not be known at all since all the required manipulations can be performed just by knowing the kernel function.
- All the computations are performed in the input space as opposed to the high (possibly infinite) dimensional feature space.

Not all functions are suitable to be used as kernel functions. In general, the function must satisfy Mercer's conditions. The details are left out here. Osuna et. al. [19, pages 11-14] provide an overview of kernels. A fairly detailed overview may be found in Burges [1]. Table 3 contains a listing of commonly used kernels.

As an aside we note that Multi Layer Perceptron (MLP) function does not satisfy Mercer conditions for all values of its parameter θ and hence may not be a kernel for certain values of θ.

Name of kernel	Formula	Parameters
Linear kernel	$K(\vec{x},\vec{y}) = \vec{x}.\vec{y}$	None.
Polynomial kernel	$K(\vec{x},\vec{y}) = (1+\vec{x}.\vec{y})^d$	d (degree of the polynomial)
Gaussian Radial Basis Function	$K(\vec{x},\vec{y}) = exp(-\frac{\lVert\vec{x}-\vec{y}\rVert^2}{2\sigma^2})$	σ
Multilayer Perceptron	$K(x,y) = tanh(\kappa\vec{x}.\vec{y}-\theta)$	$\kappa,\ \theta$

Table 3: List of commonly used kernels

Another (more general) form of the Gaussian Radial Basis function is:

$K(\vec{x},\vec{y}) = exp(-\sum_{i=1}^{N} \frac{x(i)-y(i)}{2\sigma_i{}^2})$

An equivalent form of the Gaussian Radial Basis kernel is used in the software SVM^{Light}, namely,

$$K(\vec{x},\vec{y}) = exp(-\gamma \mid\mid (\vec{x}-\vec{y}) \mid\mid^2) \tag{21}$$

B Appendix: Support Vector Machines problem formulation

In the succeeding discussion we pay attention mainly to the problem formulation. The notation followed is that in Osuna et. al. [19]. The details regarding the derivation are left out. They can be found in [19] or in any text which covers

Support Vector Machines.

QUADRATIC PROGRAM : GENERAL FORM

An optimization problem with a quadratic objective function and linear constraints is called a quadratic program. The constraints may consist of inequalities. The general quadratic program (QP) may be stated as:

$$\begin{aligned} min\ \frac{1}{2}x^T Gx + c^T x \\ s.t.\ Ax = b \\ Cx \geq d \end{aligned} \tag{22}$$

where G is a symmetric $n \times n$ matrix, $x \in \Re^n$ is a vector of unknowns, A and C are matrices of dimensions $m_a \times n$ and $m_c \times n$ respectively. c, b and d are vectors of appropriate dimensions.

If G is positive semidefinite, the problem is said to be a convex QP. In this case there is a single global minimum. Nonconvex QPs, i.e. those in which G is indefinite can be significantly more challenging since they can have several stationary points and local minima. The SVM problem formulation is a convex QP.

B.1 Linear Classifier and Linearly separable problem

Linear Classifier and Linearly separable problem [19, Section 2.3]

We wish to find the "optimal" separating hyperplane $(\mathbf{w}, b)$ such that :

$$\boldsymbol{w}.\boldsymbol{x_i} + b \geq 1 \;\; \forall \boldsymbol{x_i} \in \text{class } 1 \tag{23}$$

$$\boldsymbol{w}.\boldsymbol{x_i} + b \geq 1 \;\; \forall \boldsymbol{x_i} \in \text{class } 2 \tag{24}$$

The decision function is given by

$$f_{\mathbf{w},b} = sign(\mathbf{w}.\mathbf{x} + b) \tag{25}$$

Problem formulation

Minimize

$$\phi(\mathbf{w}) = \frac{1}{2} \mid\mid \mathbf{w} \mid\mid^2$$

subject to

$$y_i(\mathbf{w}.\mathbf{x_i} + b) \geq 1 \quad i = 1, \ldots, l \tag{26}$$

(Eqn 10. Osuna et. al [19])

B.1.1 Dual Problem

We look at the dual problem and use Lagrange multipliers since the dual problem can be extended easily to non-linear decision surfaces.

The Lagrangian is given by: (Eqn. 11 Osuna et. al. [19])

$$L(\mathbf{w}, b, \mathbf{\Lambda}) = \frac{1}{2}\|\mathbf{w}\|^2 - \sum_{i=1}^{l} \lambda_i[y_i(\mathbf{w}.\mathbf{x_i} + b) - 1] \tag{27}$$

where $\mathbf{\Lambda} = (\lambda_1, \ldots, \lambda_l)$ is the vector of non-negative Lagrange multipliers corresponding to constraints in 26

After some mathematical steps(refer Osuna et. al [19] for details) the dual quadratic program may be written as below. (Eqn. 15 in Osuna et. al)

Maximize

$$F(\mathbf{\Lambda}) = \mathbf{\Lambda}.\mathbf{1} - \frac{1}{2}\mathbf{\Lambda}.D\mathbf{\Lambda}$$

subject to

$$\mathbf{\Lambda}.y = 0$$
$$\mathbf{\Lambda} \geq 0 \tag{28}$$

where $\mathbf{\Lambda} = (\lambda_1, \ldots, \lambda_l)$ is the vector of Lagrange multipliers, $y = (y_1, y_2, \ldots y_l)$ and D is a symmetric $l \times l$ matrix with elements $D_{ij} = y_i y_j \mathbf{x_i}.\mathbf{x_j}$

CLASSIFICATION USING THE TRAINED SVM

The vectors for which $\lambda_i > 0$ are the *Support Vectors*. The bias b may be computed from any support vector as:

$$b^* = y_i - \mathbf{w}^*.\mathbf{x_i}$$

From a numerical point of view, it is better to find the bias from various support vectors and take the average. The decision function is:

$$f(\mathbf{x}) = sign(\sum_{i=1}^{l} y_i \lambda_i^* (\mathbf{x}.\mathbf{x_i}) + b^*) \tag{29}$$

where

x	=	The test vector to be classified into one of the two predefined classes ($\{$1,-1$\}$).
l	=	Number of support vectors
x_i	=	i th support vector
λ_i^*	=	Lagrange parameter of the i th support vector.
b*	=	bias

B.2 Soft margin Hyperplane:Linearly non-separable case

The dual problem may be formulated as:(Eqn 30 in Osuna et. al [19])

Maximize $F(\Lambda) = (\mathbf{\Lambda}.\mathbf{1}) - \frac{1}{2}\mathbf{\Lambda}.D\mathbf{\Lambda}$

subject to

$$\begin{aligned} \mathbf{\Lambda}.\boldsymbol{y} &= 0 \\ \mathbf{\Lambda} &\leq C\mathbf{1} \\ \mathbf{\Lambda} &\geq \mathbf{0} \end{aligned} \tag{30}$$

The decision function (equation 25) may be written as:

$$f(\mathbf{x}) = sign(\sum_{i=1}^{l} y_i \lambda_i^* (\boldsymbol{x.x_i}) + b^*) \qquad (31)$$

B.3 Nonlinear decision surfaces and kernels

Problem formulation

The quadratic programming problem is:

Maximize

$$F(\boldsymbol{\Lambda}) = \boldsymbol{\Lambda.1} - \boldsymbol{\tfrac{1}{2}\Lambda}.D\boldsymbol{\Lambda}$$

subject to

$$\begin{aligned} \boldsymbol{\Lambda.y} &= 0 \\ \boldsymbol{\Lambda} &\leq C\mathbf{1} \\ \boldsymbol{\Lambda} &\geq \mathbf{0} \end{aligned} \qquad (32)$$

where $y = (y_1, y_2, \ldots y_l)^T$ and D is a symmetric $l \times l$ matrix with elements

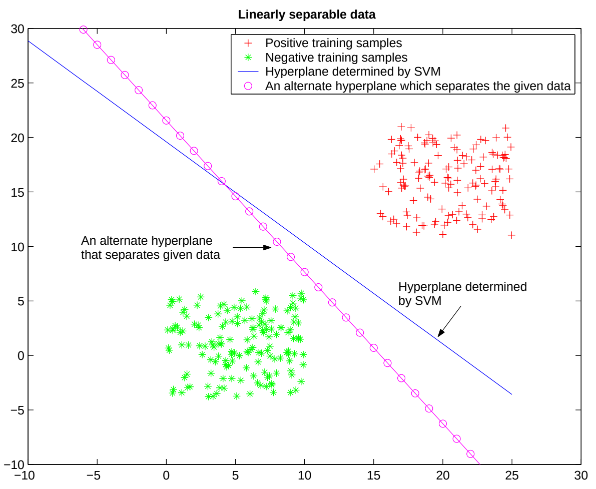
$D_{ij} = y_i y_j K(\vec{x_i}, \vec{x_j})$

Under the mapping (Equation 19) the decision function has the form :

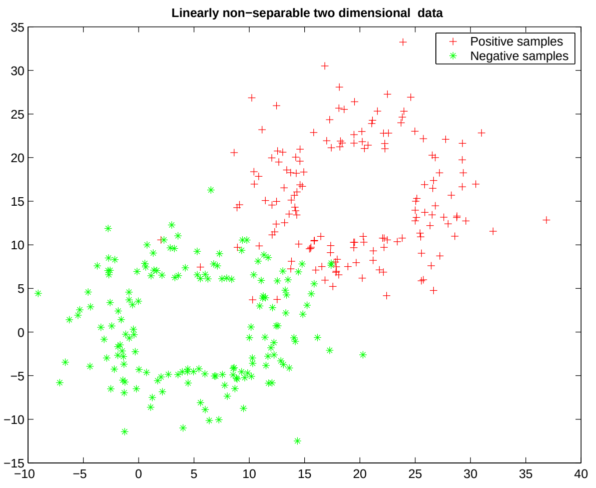
$$f(\mathbf{x}) = sign(\phi(\mathbf{x}).\mathbf{w}^* + b^*) \Rightarrow sign(\sum_{i=1}^{l} y_i \lambda_i^* \phi(\mathbf{x}).\phi(\mathbf{x_i}) + b^*) \qquad (33)$$

This can be conveniently written as:

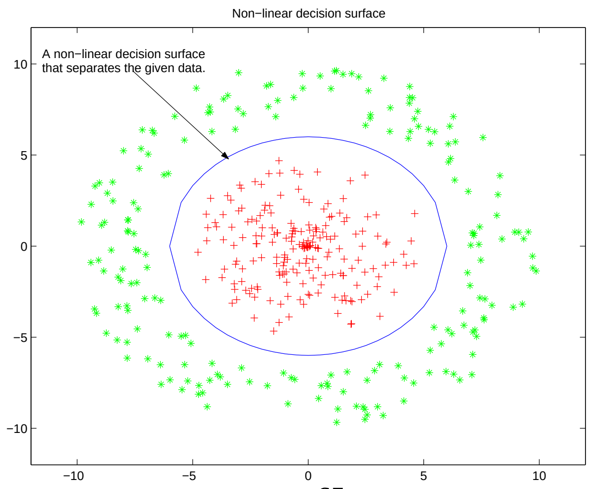
$$f(\mathbf{x}) = sign(\sum_{i=1}^{1} y_i \lambda_i^* K(\mathbf{x}, \mathbf{x_i}) + b)$$



(a)



(b)



(c)

Figure 19: (a) Linearly separable data. (b) Linearly non-separable data. (c)Non linear decision surface.